Comparing propagation models of the NS3 network simulator for 802.11n - Lab report for communication on distributed systems

B.Sc. Manuel Faatz

***Abstract*—Network simulation plays a crucial role in evaluating and testing protocols and algorithms within a controlled environment. This paper focuses on the NS3 network simulator, a widely-used open-source tool for discrete event network simulations. Specifically, the study compares various propagation loss models within NS3 for the IEEE 802.11n standard, examining their impact on wireless communication in a simple point-to-point link scenario. The selected models include FriisPropagationLossModel, FixedRssLossModel, ThreeLogDistancePropagationLossModel, TwoRayGroundPropagationLossModel, and NakagamiPropagationLossModel. To conduct the comparisons, a custom NS3 script was developed, incorporating a Minstrel HT manager, Rx power and datarate monitoring, and UdpClientServerHelper for traffic generation. Simulations were executed remotely due to computational resource requirements, and a Python wrapper facilitated remote experiment control and data analysis. The results indicate that the choice of the model significantly influences both data rate and received signal power. The study establishes a stable state after 3 seconds of simulation, providing a basis for meaningful data rate comparisons. Unexpected spikes in data rate for certain models between 50 and 100 meters warrant further investigation. The Rx power results highlight similarities between specific models, urging a deeper examination of their configurations and parameter values in future work. This research contributes valuable insights into the performance of the IEEE 802.11n standard over varying distances, emphasizing the importance of propagation model selection. It underscores the need for careful parameterization when utilizing diverse propagation models. Future work should delve into the detailed impact of model parameters on range and data rate, exploring different network topologies and interference scenarios.**

***Index Terms*—NS3, propagation loss models, 802.11n, network simulation, wireless communication**

I. Introduction

Network simulation has been a popular tool for network research for a long time. It allows testing new protocols and algorithms in a controlled environment. The NS3 network simulator is a popular open source discrete event network simulator. It is written in C++ and provides a Python interface. It is used in many research projects and is under active development. The NS3 network simulator provides a large number of models for different network layers and protocols. In order to model how signal propagation effects wireless communication, different propagation models have been developed. The NS3 network simulator provides many models for different network layers and protocols. The NS3 network simulator provides many models for different use cases and environments. In this work, some of the ones most applicable for 802.11n have been selected and compared at different distances with respect to throughput and received signal power. Distance is one of the most common factors for Rx power loss in wireless communications, since the spread of the electromagnetic waves across an increasing area with increasing distance means a lower power density at any given point of the area. Additionally, effects like reflection, refraction, absorption and interference can influence the wireless transmission performance. Depending on the propagation model, these effects are simulated in some capacity. However, there is no consensus in the scientific community on which model is most suitable for which type of simulation. Therefore, the author compared a selection of them in a simple setting to have an idea on how each of them stacks up against each other and reality. The scenario chosen for this work is a simple point to point link between two nodes. This ensures that the only factor influencing the signal propagation is the distance between the nodes and the propagation model.

II. Methodology

First, suitable propagation models had to be selected. NS3 supports a multitude of propagation models which can be found on [1, the ns3 website]. This list also includes explanations on how each model works including its equations. However for this work only the following models were deemed suitable:

- FriisPropagationLossModel
- FixedRssLossModel
- ThreeLogDistancePropagationLossModel
- TwoRayGroundPropagationLossModel
- NakagamiPropagationLossModel

In order to compare them, a ns3 script was set up. It was very similar to the script wifi-spectrum-per-example.cc with some notable changes: Instead of a constant rate Wi-Fi manager (which would make the analysis of the data rate pointless), a Minstrel HT manager was used. Furthermore, callbacks for monitoring the Rx power were added, including some tools for value conversions and averaging. Some command line arguments like simulationTime, disctance, propagationModel, etc. were added as well. The callback monitoring the data rate was implementing by dividing the app layer packet size by the time between packets. This approach had to take packet aggregation into account, since all aggregated packets have the same arrival time. Both callbacks saved the time and value of the respective variable in a std::map. At the end of the simulation, both maps were exported to csv files. This made importing them into pandas trivial. Traffic was generated at the application layer using a UdpClientServerHelper set to a constant data rate of 75 Mbps. Since these simulations required substantial computational resources, they were run on a remote server. For easy experiment control and file management, a python wrapper script was written. It allowed to run an experiment and download all the output files automatically. This wrapper was then imported into a jupyter notebook for testing and data analysis. The data analysis was then performed in the notebook using pandas and matplotlib. The source code for the ns3 script, the python wrapper and the jupyter notebooks can be found on [2, the author's github]. If you wish to reproduce the results, please follow the instructions in the README.md file on how to set up ns3.39, symlink the repositories scratch folder into ns3, set up the python environment, set up the environment variables and run the jupyter notebooks. Additionally, please refer to the NS3 documentation for more details on the ns3 script works and how to run it [3]. The expectations for the results of the different simulations were quite different. The FixedRssLossModel isolates the propagation delay as the only effect of the distance on the transmission. This means that the data rate should be constant until the distance is so large that the

propagation delay causes issues with the WiFi protocol timings. This should result in a sudden drop in the data rate as soon as this critical distance is reached. The Rx power per definition should stay constant. For the other models, the data rate should stay constant until the Rx power drops far enough that the current HtMCS value is no longer sufficient to maintain the data rate. This should result in a sudden drop in the data rate down to the datarate assiciated with the next HtMCS value. A list of the HtMCS values and their associated data rates can be found in [4, this table by the wireless lan professionals conference].

III. Results

Since the simulation contains non-deterministic processes and a steady state simulation is desired, the time until a steady state is reached first had to be determined. Initially, this was attempted by running multiple simulations with gradually increasing simulation time. The result can be seen in figure 1. However, this approach was

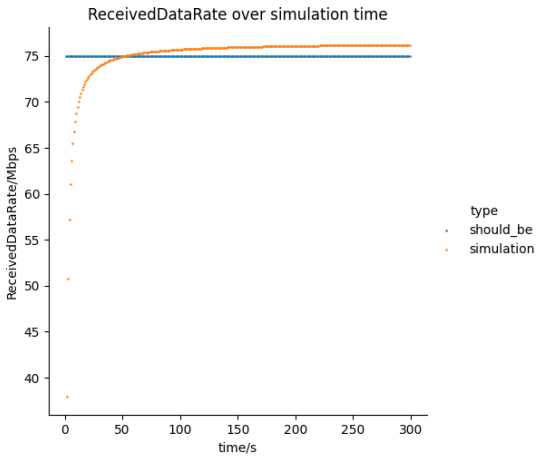

Fig. 1. average datarate over simulation duration

deemed unrealistic. Upon plotting the data rate of a single simulation over time, it was found that the data rate fluctuated heavily in the first 3 seconds and then stabilized. This can be seen in figure 2.

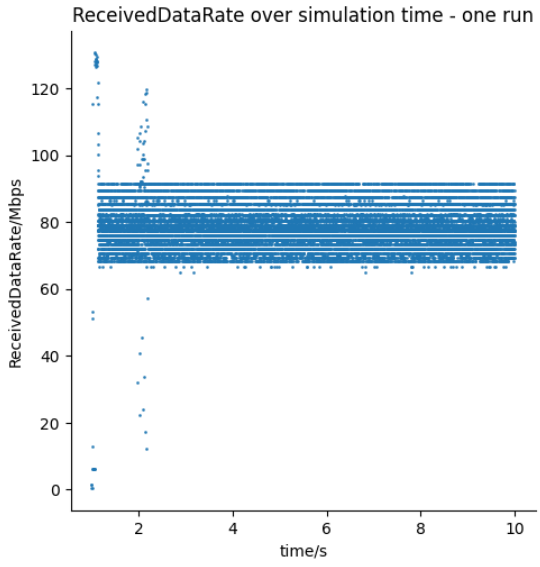

Fig. 2. datarate over simulation time of a single run

Keeping this in mind, the first graphic was redone, this time only using the data rate after 3 seconds for each simulation. The result can be seen in figure 3.

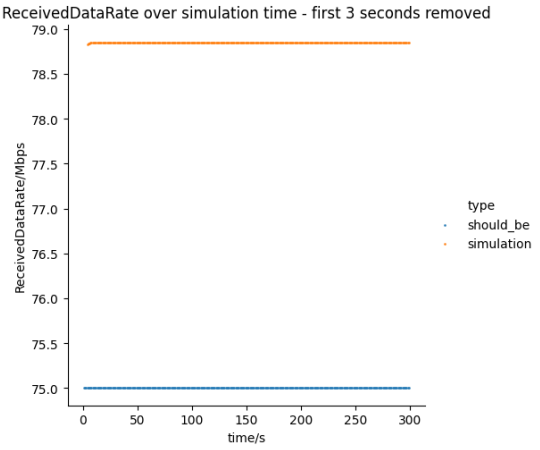

Fig. 3. average datarate over simulation duration without first 3 seconds

It shows that the average data rate of a simulation is stable after 3 seconds and independent of how much longer the simulation time is running for - which in turn means the the datarate fluctuations seen in figure 2 can be assumed to be uniformly distributed. This means as long as the simulation time is long enough after the 3 seconds to reach a statistically relevant average value, the data rate can be assumed to be independent of the simulation time. With this information, it was decided to run the simulations for 10 seconds and only use the output after 3 seconds.

The same procedure was performed for the Rx power. The results of one run can be seen in figure 4.

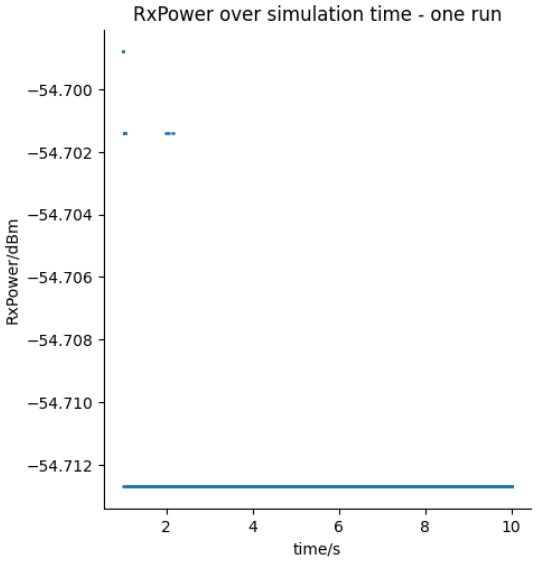

Fig. 4. Rx power over simulation duration

For the Rx power, the fluctuations are much smaller. Therefore, it was decided to simply use the average Rx power over the time of the simulation. This meant the averaging could be offloaded to the ns3 script, drastically reducing the computational load of the data analysis.

After determining the steady state, the simulations were run for 10 seconds of simulation time from a distance of 1 meter up to a

distance where no more data arrived for each propagation model with a step size of 1 meter.

The results for the data rate can be seen in figure 5. It shows that for the Friis- and the Two-Ray-Ground propagation model, the data is so similar that they are indistinguishable. This was verified by plotting each of them separately to make sure each contains data, and then plotting them together. This plot only shows informaion already contained in figure 5 and is therefore not included in this report. The results for Friis- Two-Ray-Ground- and ThreeLog-Distance propagation models also show 3 unexpexted exponential spikes in the data rate between 50 and 100 meters. The reason for this should be investigated in future work. Apart from that, the three models show the expected behaviour. The rate plateaus at a HtMCS specified rate until a critical distance is reached, after which it first drops exponentially as packets increasingly fail to reach the receiver. This behaviour stops when the acceptable limit for packet loss is reached by the remote station manager, after which it switches to a more robust HtMCS and the data rate plateaus again. This happens until a maximum distance is reached, after which no more packets arrive at the receiver. Both the Nagakami and Fixed Rss models stay fixed at a data rate of about 75 Mbps until the critical distance is reached, at which point the data rate drops to close to 0. It actually never reaches 0, but stays at between 0.1 and 0.2 Mbps. Due to limits of the computational power available, the simulations were not run for long enough to reach find out at what distance the data rate would actually reach 0. It did not at up to 4 km, and theoretically could reach over 200 km - although with a low throughput [5]. These two models were again smilar enough that they were indistinguishable in the plot. Again, separate plots were made to verify that each contained data etc.

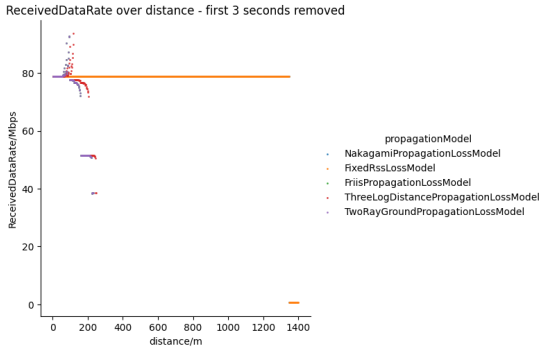

Fig. 5. data rate over distance

The results for the Rx power can be seen in figure 6.

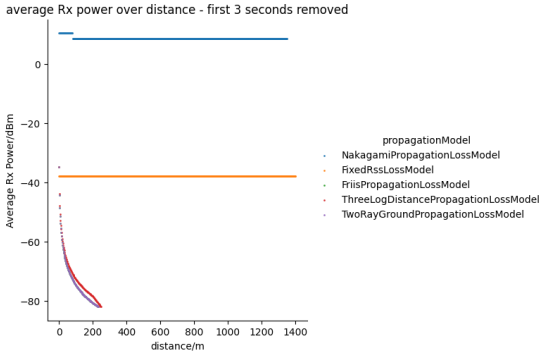

Fig. 6. Rx power over distance

The reason for the similarities between certain propagation models can be found in their configuration. Since the default values for the propagation models (as of ns3 version 3.39) were used for all models, some of the models resulted in identical equations. Future work should investigate the differences between the models with typical values for the parameters of each model.

IV. Summary

In this paper, the performance of the IEEE 802.11n standard over increasing distance using different propagation models was investigated using the NS3 (version 3.39) network simulator. It was found that for a simple ad hoc network between 2 nodes without any interference, 3 seconds suffice to reach a steady state. Additionally, the results show that the propagation model has a significant impact on the data rate and the received power. Furthermore, it was found that special care has to be taken when parametrizing the models if taking advantage of the variety of offered propagation models is desired. Future work should look at the parameters of the different propagation models and how they affect range and data rate in more detail. The influence of different topologies and interference should also be investigated. However, for some models data rates above what the application produces was observed at the appp layer between 50 and 100 meters. This should be investigated in future work. Shortly before the submission of this paper, it was noticed the ns3 script did not specify a random seed. This means that the results depend on the default random seed of the ns3 simulator and all simulations which are run with identical parameters will produce the same results. This is a common pitfall in ns3 simulations [6], and should be avoided in future work.

References

[1] NS-3. Propagation models. [Online]. Available: https://www.nsnam.org/docs/release/3.39/models/html/propagation.html
[2] M. Faatz. accompanying code. [Online]. Available: https://github.com/MainManu/KVS_ns3_lab
[3] NS-3. ns-3 documentation. [Online]. Available: https://www.nsnam.org/docs/release/3.39/
[4] W. conference. Mcs index. [Online]. Available: https://mcsindex.net/
[5] M. Rademacher, M. Chauchet, and K. Jonas, “A token-based mac for long-distance ieee802.11 point-to-point links,” 05 2016.
[6] S. Kurkowski, T. Camp, and M. Colagrosso, “Manet simulation studies: the incredibles,” *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 9, no. 4, p. 50–61, oct 2005. [Online]. Available: https://doi.org/10.1145/1096166.1096174